УДК 683.519

БЕРЕЗОВСКИЙ А.Н.
КОРОЧКИН А.В

МОДЕЛИ ВЫЧИСЛЕНИЙ ДЛЯ МАСШТАБИРУЕМЫХ КЛАСТЕРНЫХ СИСТЕМ

В работе рассматривается подход к разработке программных компонент для масштабируемых кластерных систем, основанный на использовании моделей вычислений. Предложены модели вычислений, основанные на применении теории последовательных взаимодействующих процессов Ч.Хоара. Показана возможность использования моделей для повышения качества программ и сокращения времени их разработки.

In paper the problem of creation of effective program components for scalable cluster systems based on models of computation is considered. Models of computation based on C.Hoare theory of communicating sequential processes are presented. Use models of computation for improving software and reduction design time is showed.

Кластерная архитектура сегодня рассматривается как одна из основных при построении высокопроизводительных компьютерных систем, оптимальных по критерию производительность/стоимость и включает набор вычислительных узлов, соединенных быстродействующей системой связей узлов (ССУ). Узлы кластерной системы (КС) обеспечивают обработку и хранение данных, ССУ обеспечивает быструю передачу больших объемов данных между узлами КС.

Современные КС характеризуются увеличением количества узлов, а также использованием в узлах многоядерных процессоров [1]. Кластерная система с многоядерной архитектурой (КСМА) представляет собой сложную систему, разработка программного обеспечения для которой представляет сложную задачу, связанную с организацией вычислений на двух уровнях: между узлами и внутри узлов. При этом необходимо обеспечить с помощью механизма процессов (потоков) поведение каждого узла, а также обеспечить оптимальное по времени и объемам передаваемых данных взаимодействие процессов.

Современная практика разработки приложений для сложных компьютерных систем основывается на использовании моделей вычислений, которые выступают в качестве интерфейса между архитектурой системы и программой, реализующей выбранную модель параллельного (распределенного) программирования (рис.1). Назначение модели вычислений:

- описать архитектуру КС;
- описать поведение процессов;
- описать взаимодействие процессов.

На основе модели выполнятся анализ корректности поведения процессов и их взаимодействия еще на этапе проектирования программного продукта. При этом математический аппарат модели позволяет выявить возможные причины возникновения тупиковых ситуаций. Кроме того, удачно выбранная модель может упростить процесс дальнейшего создания написания программы.

Рис.1. Модель вычислений

Кластерные системы являются хорошо масштабируемыми системами с точки зрения увеличения производительности. Кластерная архитектура допускает практически неограниченное наращивание числа узлов. Поэтому проблема разработки программного обеспечения для масштабируемых кластерных систем (МКС) является важной задачей и требует разработки методов и средств, позволяющих повысить качество приложений для МКС.

Анализ последних исследований и публикаций

Рассматриваемая тематика широко представлена в публикациях. В основном, они сводятся к выбору и обоснованию подхода к построению модели параллельных вычислений [2, 4, 5, 6, 9].

Модели вычислений могут базироваться на сетях Петри [9], теории конечных автоматов [5], CSP-теории [2, 3, 6].

Большое число разнообразных моделей вычислений не гарантирует их адекватность множеству практических задач, с которыми приходится сталкиваться при разработке распределенных приложений. Причина заключается в противоречивости требований к этим моделям и важности представления тех или иных особенностей функционирования вычислительных систем.

Нерешенные проблемы

В работе [2] предложено использование теории Ч.Хоара для построения моделей вычислений для кластерных систем. Показано, что математический аппарат CSP теории позволяет описать архитектуру кластерной системы, а также поведение и взаимодействие процессов. Однако в работе не рассмотрены вопросы построения, реализации и использования моделей вычислений на основе CSP теории для масштабируемых систем. Учитывая, что масштабируемость является важной составляющей компьютерной системы, решение проблем, связанных с созданием программного обеспечения для МКС, является актуальной задачей.

Постановка задачи

В работе [4] выделены две особенности распределенных вычислений в МКС, которые определяют требования к их разработке. Это *недетерминизм,* означающий произвольные последовательности формирования компонентов структурированных данных и *неоднородность,* которая связана с произвольной структурированностью данных, которыми обмениваются взаимодействующие процессы.

Недетерминизм связан с обеспечением однозначности результатов вычислений, при которой одному входному набору данных соответствует один выходной набор результатов. Неоднородность сообщений связана с тем, что процессы могут принимать (передавать) данные произвольного размера, зависящего от количества узлов в системе. Выбор средств для построения моделей вычислений должен обеспечить решение указанных проблем.

Целью настоящей работы является развитие подхода, предложенного в работе [2], для создания и использования моделей вычислений при разработке программных компонент для масштабируемых кластерных систем с многоядерной архитектурой.

Модели вычислений

При разработке масштабированных распределенных программ необходимо обеспечить возможность статического (динамического) создания (порождения) множества процессов, описание их поведения, запуск на выполнение и организацию взаимодействия.

Существуют различные средства описания процессов в современных языках и библиотеках параллельного программирования [7]. Языки программирования Java, Ада позволяют описать процесс с помощью специальных программных модулей (классов). В языке C# и библиотеке Win32 для этого используются потоковые функции. OpenMP позволяет выделять параллельные участки программы с помощью операторов `parallel/end parallel`. Библиотека MPI создает параллельные процессы как копии основной программы.

Программа для масштабируемой архитектуры должна базироваться на возможности создания (как статически, так и динамически) любого множества процессов. Это возможно только с помощью тиражирования базового процесса, параметризация которого позволит описать все различия между создаваемыми копиями этого процесса. В языках, поддерживающих объектно-ориентированную парадигму, это можно сделать с помощью классов и конструкторов, в языка Ада для этого используется задачный тип с дискриминантом, в MPI – идентификатор задачи, автоматически порождаемый при создании копии процесса.

Организация взаимодействия процессов включает синхронизацию процессов и обмен данными [6]. Модели взаимодействия при этом базируются на общих переменных или на посылке сообщений.

При использовании общих переменных возникают задачи взаимного исключения и

синхронизации, решение которых надо обеспечить для произвольного числа процессов. Для задачи взаимного исключения это не представляет сложности, так как примитивы типа семафоры, мютексы, критические секции, также мониторы обеспечивают контроль общих ресурсов для любого числа процессов, которые их используют. Проблемы могут возникнуть при решении задачи синхронизации, так как здесь имеют место сложные формы синхронизации вида: *процесс→группа процессов, группа процессов→процесс, группа процессов→ группа процессов.* При использовании низкоуровневых примитивов потребуется динамическое создание и использование объектов синхронизации. Эта проблема решается путем использования мониторных объектов синхронизации, которые позволяют эффективно реализовать указанные формы групповой синхронизации.

При использовании механизма посылки сообщений в МКС основная проблема связана с формированием объема передаваемых данных, который может меняться в зависимости от количества динамически создаваемых процессов. Здесь важной является возможность организация обмена данными произвольной размерности. В MPI это обеспечивается функциями `MPI_Gatherv`, `MPI_Scat-terv`, в языке Ада – использованием типов данных с уточняемыми границами, что позволяет описать вход задачи для передачи любого объема данных.

Кроме того, эффективность взаимодействия увеличивается, если язык (библиотека) поддерживают коллективные формы взаимодействия, как при передаче сообщений, так и при синхронизации. В этом плане представляют интерес коллективные операции MPI, которые реализованы в виде операций групповой рассылки (сборки) данных `MPI_Bcast`, `MPI_Gather`, `MPI_Scatter` и операций групповой синхронизации `MPI_Barrier`.

Таким образом, можно сформулировать основные требования к моделям вычислений для масштабируемых систем: поддержка недетерминизма для неограниченного числа динамически порождаемых процессов, взаимодействующих путем посылки сообщений произвольного объема и через групповую синхронизацию.

Рассмотрим возможности теории Ч.Хоара с точки зрения реализации сформулированных требований. Теория Ч.Хоара позволяет описать поведение последовательных недетерминированных процессов с произвольным независимым чередованием процессов. Это позволяет описать особенности работы МСК. Поддерживается динамическое порождение процессов без ограничения их количества. Обеспечивается синхронная однонаправленная без буферизации посылка сообщений. При этом рассматривается одна форма, когда взаимодействую только две задачи, то есть не используются групповые операции обмена. Не рассматривает общие переменные как средства взаимодействия процессов, тем более для групповой синхронизации и взаимного исключения.

Таким образом, теория Ч.Хоара может быть использована для построения моделей вычислений для МКС при ее расширении (доработке) в плане реализации двунаправленных механизмов посылки сообщений произвольной размерности, а также мониторподобных средств для взаимного исключения и синхронизации при описании группового взаимодействия процессов.

Модели вычислений для МКС

Рассмотрим МКС как сложную динамическую систему, структура которой может меняться как путем изменения числа узлов и связей между узлами. В общем случае, любая кластерная система является полносвязной, так как любые два узла связаны между собой и допускаю прямую передачу данных между собой. В этом случае масштабирование реализуется простым добавлением новых узлов. Если архитектура МКС построена путем объединения узлов в группы (подкластеры), то взаимодействие осуществляется между подкластерами и внутри каждого подкластера. При этом возможно использование различных коммуникационных систем для взаимодействия между подкластерами и внутри каждого подкластера.

Рассмотрим использование теории Ч.Хоара для построения моделей вычислений для масштабируемых кластерных систем с многоядерной архитектурой, в узлах которых используются многоядерных процессоры. Масштабированность КС на структурном уровне связана с возможностью изменения количества узлов и количества ядер

в многоядерных процессорах в узле системы. Теория Ч.Хоара позволяет поведение любой системы рассматривать в терминах объект, абстрагируясь при этом от аппаратной или программной сущности системы построении модели вычислений. Это позволяет упростить переход от описания аппаратной части к программной реализации модели.

При разработке модели вычислений будем исходить из того, что она должна быть двухуровневой и соответствовать двум уровням структуры МКС. Модель *МКС1* описывает поведение системы, связанное с взаимодействием узлов, а модель *МКС2* описывает поведением системы внутри каждого узла.

Модель первого уровня МКС1. Для модели *МКС1* используем модель клиент-сервер, которая включает один объект клиент и P объектов серверов. Параметр P может меняться, тем самым обеспечивая масшабируемость на множестве узлов.

Набор событий, связанных с объектом клиент K можно представить в виде:

$$\alpha K = \{K.\text{ввод}, K.\text{вывод}, K.\text{принять}.C, K.\text{передать}.C\},$$

где события: *К.ввод* – ввод данных на клиенте, *К.вывод* – вывод результат на клиенте, *К.принять.С* – групповой прием данных от всех P серверов C_i, ($i = 1..P$), *К.передать.С* – групповая передача данных всем серверам C_i. Масштабируемость здесь связана с групповыми передачами и приемами данных между клиентом и всеми серверами. Для реализации групповых операций *К.принять.С* и *К.передать.С* их необходимо параметризировать через параметр P. Кроме того, эти операции должны позволять либо передачу любого объема данных, либо допускать настройку на конкретный объем передаваемых данных. В MPI при передаче данных используются функции `Send/Receive`, в которых указывается объем передаваемых данных. В языке Ада - типы с уточняемыми границами (`type Vector is array(integer range <>) of float`), позволяющие передавать объемы данных любой размерности [8].

Набор событий, связанных с i-м объектом сервер (C_i, $i = 1 .. P$) можно представить в виде

$$\alpha C_i = \{ C_i.\text{принять}.K, C_i.\text{передать}.K, C_i.\text{счет} \}$$

где события: C_i*.принять.К* – прием данных от клиента, C_i*.передать.К* - передача данных клиенту, C_i*.счет* - счет на сервере. Масштабируемость на серверной стороне не фиксируется.

Поведение объекта клиент, используя алфавит αK, можно описать как процесс

$$K = (K.\text{ввод} \rightarrow K.\text{передать}.C \rightarrow K.\text{принять}.C \rightarrow K.\text{вывод}).$$

Для множества объектов клиентов C_i их поведение описывается следующим процессом

$$C_i = (C_i.\text{принять}.K \rightarrow C_i.\text{счет} \rightarrow C_i.\text{передать}.K).$$

Модель *МКС1* можно описать как параллельную комбинацию поведения процессов K и C_i :

$$\text{СЕРВЕРЫ} = (C_1 \parallel C_2 \parallel \ldots \parallel C_p),$$
$$\text{КЛИЕНТ} = (K),$$
$$\text{МКС1} = (\text{КЛИЕНТ} \parallel \text{СЕРВЕРЫ}).$$

Взаимодействие процессов в модели МКС1. Взаимодействие процессов в модели *МКС1* осуществляется на основе посылки сообщений. Для описания взаимодействия процессов используем объект коммуникации (*ОК*), предложенный в работе [2]. Взаимодействие через *ОК* реализуется как обращение к объекту (вызов *ОК* и прием вызова *ОК*) :

$$\alpha OK = \{ OK.\text{передать}, OK.\text{принять} \}.$$

Процесс, который описывает поведение объекта коммуникации, задается рекурсивным соотношением:

$$OK = ((OK.\text{принять} \mid OK.\text{передать}) \rightarrow OK).$$

ОК является параметризированным объектом *ОК(Д, Н, В, Б)*, где параметр *Д* определяет тип и объем передаваемых данных, параметр *Н* - направление передаваемых данных, параметр *В* - синхронное или асинхронное взаимодействие, параметр *Б* – буферизированное или не буферизованное взаимодействие. Для реализации масштабированности параметр *Д* должен допускать возможность указания объема передаваемых данных (*Д.Х*), зависящего от P.

В теории Ч.Хоара поведение объекта *ОК* описывается процессом, который является подчиненным для процессов, которые через него взаимодействуют. Корректность такого взаимодействия определяется следующими правилами:

Правило 1 определяет необходимое условие взаимодействия процессов через *ОК* относительно алфавитов процессов: “Если A и

B процессы, *Z* - объект коммуникации, то необходимым условием взаимодействия *A* и *B* через *Z* есть выполнение соотношения $\alpha Z \subseteq (\alpha A \cup \alpha B)$ ".

Правило 2 определяет необходимое условие взаимодействия процессов через *ОК* относительно поведения процессов: "Если *A* и *B* процессы, *Z* - объект коммуникации, то необходимым условием взаимодействия *A* и *B* через *Z* есть следующие действия в процессах $Z = ((Z.принять \mid Z.передать) \rightarrow Z)$, $A = ((Z.передать) \rightarrow A)$, $B = ((Z.принять) \rightarrow B)$, $(Z \;//\; (A \parallel B))$. При этом $A.Z.Д.X = B.Z.Д.X$".

Реализация объекта коммуникации для взаимодействия узлов МКС можно выполнить различными способами. С учетом сетевой технологии, которая определяет аппаратное взаимодействие узлов, это могут быть сокеты, а также удаленные методы [6]. При использовании удаленных методов для операции умножения вектора на матрицу $A = B*MC$ в МКС, включающей *P* серверов, каждый сервер содержит удаленную процедуру `Счет()`, реализующую вычисления части результата $A_н=B*MC_н$, где H = N/P.

Описание удаленной процедуры на стороне сервера:

```
type Vector is array(integer
                  range    <>)    of
float;
type Matrix is array(integer
          range    <>)    of    Vec-
tor(1..N);
procedure Счет(B: in Vector; MC: in
             Matrix;     A:     out
Vector);
```

Удаленный вызов на стороне клиента для *i*-го сервера:

```
Сервер.Счет(B(1..N),MC(h*(i-1)+1..
             i*H),A(h*(i-
1)+1..i*H);
```

Модель второго уровня МКС2. Для модели *МКС2,* описывающей поведение узла, используем модель мастер-рабочий. Объект мастер отвечает за связь с клиентским узлом МКС и здесь связываются первый и второй уровни системы. Объект мастер (*B*) принимает исходные данные от клиента, распределяет их в узле по всем *K* объектам рабочий, собирает результаты счета в узле и отсылает результат клиенту. Объект рабочий (*W*) принимает данные от объекта мастер, выполняет счет и возвращает результат мастеру.

Набор событий, связанных с объектом мастер *i*-го узла (B_i), можно представить в виде

$\alpha B_i = \{B_i.принять.W_{ij}, B_i.передать.W_{ij}, B_i.счет\}$,

где события: $B_i.передать.W_{ij}$ – передать данные в узле объекту рабочий W_{ij}, $B_i.счет$ – счет, $B_i.принять.W_{ij}$ – принять данные в узле от объекта рабочий W_{ij}, ($j = 1..K$, K – количество ядер в многоядерном процессоре).

Набор событий, связанных с объектом *j*-й рабочий *i*-го узла (W_{ij}) можно представить в виде

$\alpha W_{ij} = \{ W_{ij}.принять.B_i, W_{ij}.передать.B_i, W_{ij}.принять.W_{iк}, W_{ij}.передать.W_{iк}\}$,

где события: $W_{ij}.принять.B_i$ – прием данных от мастер объекта, $W_{ij}.передат.B_i$ – передача данных мастер объекту, $W_{ij}.счет$ – счет в объекте рабочий, $W_{ij}.принять.W_{iк}$ – прием данных от объекта рабочий $W_{iк}$, $W_{ij}.передать.W_{iк}$ – передача данных объекту рабочий $W_{iк}$.

Описание поведения объектов мастер и рабочий зависит от системы связей ядер в многоядерном процессоре, которая будет определять форму взаимодействия объектов мастер и рабочий. В общем случае для полносвязной архитектуры узла, когда все ядра связаны между собой, поведение объектов в узле на основе модели посылки сообщений можно представить следующим образом.

Поведение объекта мастер B_i (процесс мастера) используя алфавит αB_i, можно описать как

$B_i = (B_i.передать.W_{ij} \rightarrow B_i.счет \rightarrow B_i.принять.W_{ij})$.

Поведение объекта рабочий W_{ij}:

$W_{ij} = (W_{ij}.принять.B_i \rightarrow W_{ij}.счет \rightarrow W_{ij}.передать.B_i)$.

Модель второго уровня *МКС2* для *i*-го клиентского узла можно описать как параллельную комбинацию поведения процессов B_i и W_{ij}:

$$МАСТЕР_i = (B_i), \quad РАБОЧИЕ_i = (W_i),$$
$$УЗЕЛ_i = (B_i \parallel РАБОЧИЕ_i).$$

Взаимодействие процессов в модели МКС2. При рассмотрении вопросов масштабирования внутри узлов МКС, построенных с использованием многоядерных процессоров, взаимодействие процессов осуществля-

ется на основе общей памяти и модели общих переменных. Проблема масштабирования в узле МКС характерна для SMP – систем и связана с наличием общей памяти и когерентностью кэш-памяти.

В работе [2] введено понятие объекта синхронизации (*ОС*) для организации взаимодействия через общие переменные. Для реализации групповой синхронизации будем использовать объект синхронизации типа М, который реализует концепцию мониторов. При этом необходимо дополнить *ОС(М)* средствами, обеспечивающими эффективную реализацию групповой синхронизации.

Набор событий, связанных с параметризированным объектом *ОС*(*М)* для синхронизации можно представить в виде

$\alpha M(к)$ = { *М.ждать*(*к*), *М.сигнал*, *М.несигнал*, *М.установить(к)* },

где события: *М.ждать(к)* – ожидание сигнала от *к* процессов, *М.сигнал*– сигнал о событии, *М.несигнал* – установка *ОС(М)* в несигнальное состояние, *М.установить* – настроить объект на синхронизацию с *к* процессами.

В задаче групповой синхронизации, когда один процесс (А) ждет сигналов от *s* процессов (E_1 .. E_s), необходимо поведение *ОС(М)* описать как *М* = ((*М.несигнал*) → *М.установить(s)* →((*М.ждать(s)* | *М.сигнал*) → *М*)), ожидание сигналов в процесс *А* описать как *М.ждать(s),* а посылку сигналов в процессах E_1 .. E_s реализовать событием *М.сигнал.*

Использование моделей вычислений

После построения модели вычислений она может быть использована для анализа корректности взаимодействия процессов, который выполняется на основе проверки предложенных правил. Это позволит выявить и предотвратить тупиковые ситуации, оптимизировать объемы передаваемой информации, выбрать более эффективные средства коммуникации и синхронизации.

Второй важной стороной моделей вычислений является возможность их применения при разработке распределенных приложений. Использование моделей позволяет упростить построение алгоритмов процессов и их последующую программную реализацию. Построение алгоритмов процессов осуществляется в терминах событий, которые были определены и использовались для объектов моделей. Сами алгоритмы полностью соответствуют поведению объектов в модели вычислений. При реализации модели следует учитывать тот факт, что большинство современных языков параллельного программирования реализуют концепцию взаимодействующих последовательных процессов, составляющих основу теории Ч.Хоара. Это упрощает реализацию процессов для объектов с помощью процессов (потоков) в программе.

Выбор и реализация объектов коммуникации и синхронизации также упрощается, так как они разработаны с учетом и анализом различных существующих механизмов взаимодействия процессов, таких как семафоры, мютексы, критические секции, замки, мониторы, рандеву [7].

Авторами были проведены экспериментальные исследования эффективности использования предложенных моделей вычислений, построенных на основе теории Ч.Хоара, для разработки приложений для реальной масштабируемой кластерной системы, оснащенной четырехядерными процессорами. Модели вычислений позволили описать особенности архитектуры КСМА, включающей переменное число узлов. Использование моделей вычислений для разработки распределенных приложений позволило сократить время их программирования и отладки. Использовались языки программирования C, Java, C#, Ада, библиотеки Win32, MPI. Разработанный пакет программ эффективно выполнялся в МКС с любым числом узлов, обеспечивая высокоскоростную передачу данных между узлами и высокий коэффициент загрузки многоядерных процессоров в узлах.

Выводы

Предложен подход к построению моделей вычислений для масштабируемых кластерных систем с многоядерной архитектурой, основанный на расширении теории взаимодействующих последовательных процессов путем введения параметризованных объектов синхронизации и коммуникации. Модели позволяют описать архитектуру кластерной системы, поведение процессов и их взаимодействие с учетом особенностей масштабированности системы. Математический аппарат CSP теории позволил сформулировать ряд требований для оптимального взаимо-

действия процессов, выполнить анализ корректности взаимодействия процессов на обоих уровнях кластерной системы при использовании модели, основанной на общих переменных, и модели, основанной посылке сообщений.

Использование предложенных моделей при проектировании программных компонент МКС позволит повысить качество программ и сократить время их разработки.

Список литературы

1. TOP500 Supercomputing sites. http://www.top500.org
2. Садег Растгу. Модели вычислительных процессов в кластерных системах с многоядерной архитектурой // Вісник НТТУ "КПІ".Сер. Інформатика, управління та обчислювальна техніка. № 48. – 2008. С.58-62.
3. Хоар Ч.Взаимодействующие последовательные процессы: Пер. с англ. – М.: Мир, 1989 .– 264 с.
4. Топорков В.В. Модели распределенных вычислений. – М.: ФИЗМАТЛИТ, 2004. – 320 с.
5. Любченко В.С. Автоматная модель параллельных вычислений / В.С.Любченко // Высокопроизводительные параллельные вычисления на кластерных системах : 3-я межд. конф. 2-4 окт. 2006 г. : труды конф. – М. - 2006. – С. 1359 – 1374.
6. Березовский А. Организация вычислений в масштабируемых кластерных системах // Комп'ютерні системи та мережеві технології: друга міжнар.наук.- техн.конф., CSNT'2009, 10-12 черв. 2009 р.: збірник наукових праць. – 2009, вип. 2(24).
7. Жуков І.А., Корочкін О.В. Паралельні та розподілені обчислення. Навч. посібник. – К.: Корнійчук, 2005. – 226 с.
8. Корочкин А.В. Ада95. Введение в программирование. – К.: Корнійчук, 1998. – 240 с.
9. Каляев И.А. Реконфигурируемые мультиконвейерные вычислительные структуры / И.А.Каляев, И.И.Левин, Е.А.Семерников, В.И.Шмойлов. – М.: ид-во ЮНЦ РАН, 2008. – 320 с.

Поступила в редакцию 14.12.2009